Arquitectura de Computadores

Santiago Ramirez Arenas
Docente: José Alfredo Jaramillo Villegas
Universidad Tecnológica de Pereira

15 de octubre de 2021

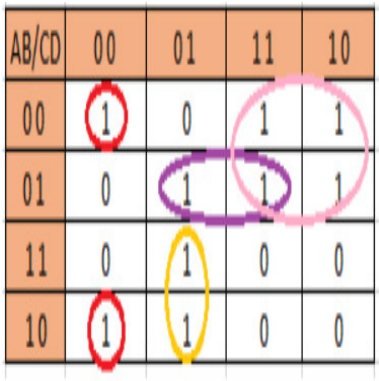
Problemas de Lógica Combinatoria.
Este laboratorio está diseñado para reforzar los conocimientos sobre lógica combinatoria, para un mayor desempeño y entendimiento de problemas lógicos. Se basa en la interpretación y simplificación de circuitos lógicos.

Circuito combinatorio de 4 a 1 bits

A	B	C	D	S
0	0	0	0	1
0	0	0	1	0
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	0
1	0	0	0	1
1	0	0	1	1
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	1
1	1	1	0	0
1	1	1	1	0

Diseñar la función lógica del circuito.
Simplificar el circuito por mapas de Karnaugh.
Implementar una simplificación del circuito por medio de una ecuación booleana.
Implementar una simplificación del circuito con llamados a módulos de compuertas lógicas.

Mapa de karnaugh



AB/CD	00	01	11	10
00	1	0	1	1
01	0	1	1	1
11	0	1	0	0
10	1	1	0	0

Simplificando el mapa de Karnaugh nos da la siguiente Ecuación Booleana:

$$S\ (A, B, C, D) = A'C + A'B'D' + BC'D + AB'C'$$

-Implementación usando Ecuación Booleana y llamado a Módulos de Compuerta Lógicas

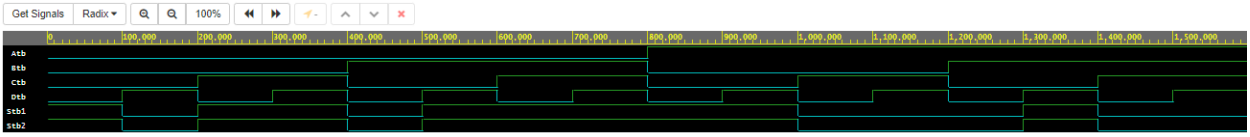
```
`timescale 1ns / 100ps
 module ejercicio1(
  input A,B,C,D,
  output S1,S2);
  wire T,U,V,W,X,Y;

  assign S1  = ((!A & C) | (!A & !B & !D) | (B & !C & D) | (A & !B & !C));

  and comp1 (T,!A,C);
  and comp2 (U,!A,!B,!D);
  and comp3 (V,B,!C,D);
  and comp4 (W,A,!B,!C);
  or  comp5 (X,T,U);
  or comp6  (Y,V,W);
  or comp7 (S2,X,Y);
endmodule
```

Testebench Llamado a módulo de compuertas lógicas y Ecuaciones lógicas

```
`timescale 1ns / 100ps
module test_ejercicio1 ();
    logic Atb=0;
    logic Btb=0;
    logic Ctb=0;
    logic Dtb=0;
    logic Stb1;
    logic Stb2;
    integer i=1;

 ejercicio1 circuito (
   .A(Atb),
   .B(Btb),
```

```
14     .C(Ctb),
15     .D(Dtb),
16     .S1(Stb1),
17     .S2(Stb2));
18
19      initial begin
20      $dumpfile("dump.vcd");
21        $dumpvars(1,test_ejercicio1);
22        for(i=1;i<17;i=i+1)
23        begin
24        #100
25          if(i==8)begin
26          Atb=1;
27        end
28          if(i%4==0)begin
29          Btb=~Btb;
30        end
31          if(i%2==0)begin
32          Ctb=~Ctb;
33        end
34        Dtb=~Dtb;
35      end
36     $finish;
37   end
38
39 endmodule
```

Simulación S1 y S2 usando Ecuación Booleana y Llamado a Módulso de Compuertas Lógica



Segundo ejercicio Función lógica 5 a 1 bits.

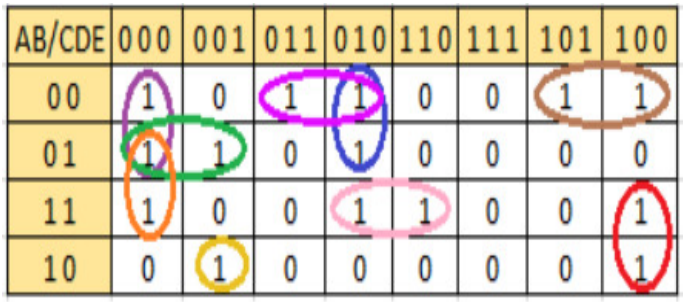
$$f(a, b, c, d, e) = \text{Sigma}(0, 2, 3, 4, 5, 8, 9, 10, 17, 20, 24, 26, 28, 30).$$

Hacer tabla de verdad del circuito.
Simplificar por mapas de Karnaugh.
Implementar una simplificación usando una ecuación booleana.
Implementar una simplificación con llamados a módulos de compuertas lógicas

Tabla de verdad

A	B	C	D	E	S
0	0	0	0	0	1
0	0	0	0	1	0
0	0	0	1	0	1
0	0	0	1	1	1
0	0	1	0	0	1
0	0	1	0	1	1
0	0	1	1	0	0
0	0	1	1	1	0
0	1	0	0	0	1
0	1	0	0	1	1
0	1	0	1	0	1
0	1	0	1	1	0
0	1	1	0	0	0
0	1	1	0	1	0
0	1	1	1	0	0
0	1	1	1	1	0
1	0	0	0	0	0
1	0	0	0	1	1
1	0	0	1	0	0
1	0	0	1	1	0
1	0	1	0	0	1
1	0	1	0	1	0
1	0	1	1	0	0
1	0	1	1	1	0
1	1	0	0	0	1
1	1	0	0	1	0
1	1	0	1	0	1
1	1	0	1	1	0
1	1	1	0	0	1
1	1	1	0	1	0
1	1	1	1	0	1
1	1	1	1	1	0

Mapa de Karnaugh



Ecuación Booleana

(A, B, C, D, E) = A'B'C'D + A'B'CD' + A'BC'D' + AB'C'D'E + ABE' + A'C'E' + B'CD'E' Mapa de Karnaugh

-Implementación usando Ecuación Booleana y llamado a Módulos de Compuerta Lógicas

```
`timescale 1ns / 100ps
 module ejercicio2(
  input A,B,C,D,E,
  output S1,S2);
  wire T,U,V,W,X,Y,Z;

   assign S1  = ((!A & !B & !C & D) | (!A & !B & C & !D) | (!A & B & !C &
   !D) | (A & !B & !C & !D & E) | (A & B & !E) | (!A & !C & !E) | (!B & C
   & !D & !E));
```

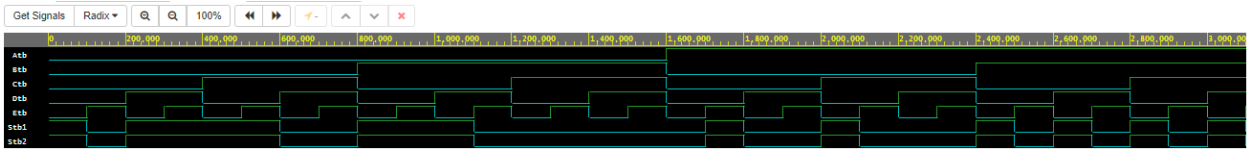
```
8
9    and comp1 (T,!A,!B,!C,D);
10   and comp2 (U, !A,!B,C,!D);
11   and comp3 (V,!A,B,!C,!D);
12   and comp4 (W,A,!B,!C,!D,E);
13   and comp5 (X,A,B,!E);
14   and comp6 (Y,!A,!C,!E);
15   and comp7 (Z,!B,C,!D,!E);
16
17   or comp8 (S2,T,U,V,W,X,Y,Z);
18 endmodule
```

Testebench Llamado a módulo de compuertas lógicas y Ecuaciones lógicas

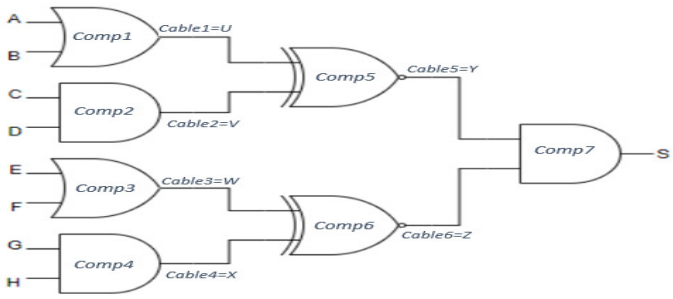
```
1  `timescale 1ns / 100ps
2  module test_ejercicio2 ();
3      logic Atb=0;
4      logic Btb=0;
5      logic Ctb=0;
6      logic Dtb=0;
7      logic Etb=0;
8      logic Stb1;
9      logic Stb2;
10     integer i=1;
11
12  ejercicio2 circuito (
13    .A(Atb),
14    .B(Btb),
15    .C(Ctb),
16    .D(Dtb),
17    .E(Etb),
18    .S1(Stb1),
19    .S2(Stb2));
20
21     initial begin
22     $dumpfile("dump.vcd");
23       $dumpvars(1,test_ejercicio2);
24       for(i=1;i<32;i=i+1)
25       begin
26       #100
27         if(i==16)begin
28         Atb=1;
29       end
30         if(i%8==0)begin
31         Btb=~Btb;
32       end
33         if(i%4==0)begin
34         Ctb=~Ctb;
35       end
36         if(i%2==0)begin
37         Dtb=~Dtb;
38       end
39       Etb=~Etb;
40     end
41    $finish;
```

```
  end
 endmodule
```

Simulación S1 y S2 usando Ecuación Booleana y Llamado a Módulso de Compuertas Lógica



- Circuito lógico de 8 entradas a 1 salida.



Implementar una simplificación usando una ecuación booleana.
Implementar una simplificación con llamados a módulos de compuertas lógicas reducidas (And y Or).

-Implementación usando Ecuación Booleana y llamado a Módulos de Compuerta Lógicas

```
`timescale 1ns / 100ps

 module ejercicio3(
  input A,B,C,D,E,F,G,H,
  output S1,S2);
  wire T,U,V,W,X,Y,Z,S,O,P,R,Q, S3, S4;


   assign S2 =(((A|B)~^ (C & D))&((E | F)~^(G & H)));

   or comp1 (T,A,B);
   and comp2 (U,C,D); //
   or comp3 (V,E,F);
   and comp4 (W,G,H);//

   //Primera xnor
   nand comp6 (X,T,U);
   nand comp7 (Y,X,T);
   nand comp8 (Z,X,U);
   nand comp9 (S,Y,Z);
   nand comp10 (S3,S);
   //Segunda  xnor
   nand comp11 (O,V,W);
```

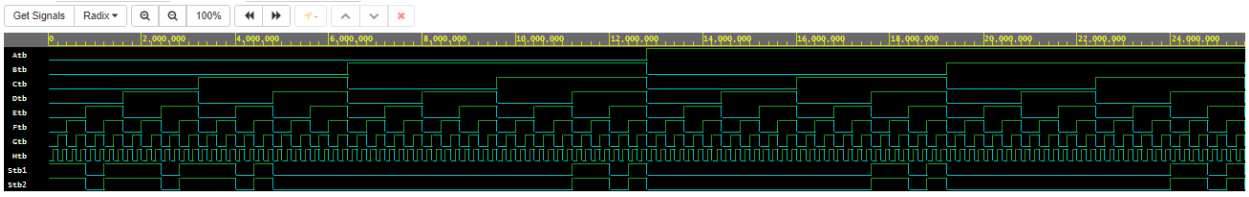
```
  nand comp12 (P,O,V);
  nand comp13 (R,O,W);
  nand comp14 (Q,P,R);
  nand comp15 (S4,Q);
  // Ultima
  and comp16 (S1,S3,S4);
endmodule
```

Testebench Llamado a módulo de compuertas lógicas y Ecuaciones lógicas

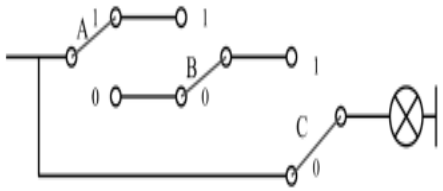
```
module  test_ejercicio3;

logic Atb=0;
logic Btb=0;
logic Ctb=0;
logic Dtb=0;
logic Etb=0;
logic Ftb=0;
logic Gtb=0;
logic Htb=0;
logic Stb1;
logic Stb2;
integer i=1;

ejercicio3 circuito (
   .A(Atb),
   .B(Btb),
   .C(Ctb),
   .D(Dtb),
   .E(Etb),
   .F(Ftb),
   .G(Ftb),
   .H(Ftb),
  .S1(Stb1),
  .S2(Stb2));

initial begin
   $dumpfile("dump.vcd");
   $dumpvars(1,test_ejercicio3);
   for(i=1;i<257;i=i+1)
     begin
     #100
       if(i==128)begin
       Atb=1;
     end
       if(i%64==0)begin
       Btb=~Btb;
     end
       if(i%32==0)begin
       Ctb=~Ctb;
     end
       if(i%16==0)begin
       Dtb=~Dtb;
     end
       if(i%8==0)begin
```

```
46            Etb=~Etb;
47        end
48          if(i%4==0)begin
49            Ftb=~Ftb;
50        end
51          if(i%2==0)begin
52            Gtb=~Gtb;
53        end
54        Htb=~Htb;
55      end
56     $finish;
57    end
58  endmodule
```

Simulación S1 y S2 usando Ecuación Booleana y Llamado a Módulso de Compuertas Lógica



Circuito de una bombilla de 3 entradas.



Hacer tabla de verdad del circuito.
Implementar una simplificación usando una ecuación booleana.
Implementar una simplificación con llamados a módulos de compuertas lógicas.
Tabla de verdad

A	B	C	S
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

Ecuación Booleana: S = (A'B'C')+(A'B'C)+(A'BC')+(AB'C')+(ABC')+(ABC)

-Implementación usando Ecuación Booleana y llamado a Módulos de Compuerta Lógicas

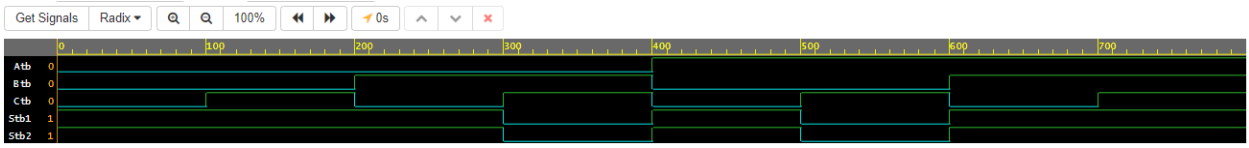
```
module ejercicio4(
 input logic A,B,C,
 output logic S1,S2
);
  wire U,V,W,X,Y,Z;

  assign S1 = ((!A & !B & !C)|(!A & !B & C)|(!A & B & !C)|(A & !B &
   !C)|(A & B & !C)|(A & B &C));

  and comp1 (U,!A,!B,!C);
  and comp2 (V,!A,!B,C);
  and comp3 (W,!A,B,!C);
  and comp4 (X,A,!B,!C);
  and comp5 (Y,A,B,!C);
  and comp6  (Z,A,B,C);
  or  comp7 (S2,U,V,W,X,Y,Z);

endmodule
```

Testebench Llamado a módulo de compuertas lógicas y Ecuaciones lógicas

```
 module test_ejercicio4;
 logic Atb =0;
 logic Btb =0;
 logic Ctb =0;
 logic Stb1 ;
 logic Stb2 ;
 integer i =1;

 ejercicio4 circuito(
 .A(Atb),
 .B(Btb),
 .C(Ctb),
 .S1(Stb1),
 .S2(Stb2)
 );

 initial begin
 $dumpfile ("dump .vcd") ;
 $dumpvars (1,test_ejercicio4);

 for (i =1; i <9; i = i +1)
 begin
 #100
  if(i==4) begin
   Atb =1;
 end
   if(i%2==0) begin
   Btb = ~Btb;
 end
   Ctb = ~Ctb;
 end
 $finish;
```

```
 end
endmodule
```

Simulación S1 y S2 usando Ecuación Booleana y Llamado a Módulso de Compuertas Lógica



El área de mantenimiento de la UTP quiere fabricar un robot para hacer el trabajo riesgoso del campus, para eso le solicitan ayuda a los estudiantes de ingeniería para implementar un módulo en el que ingresa una señal con el respectivo id de la operación que va a ejecutar (girar a la derecha o Izquierda, Avanzar, Retroceder, detenerse, elevar manos, bajar manos). Retornar una cadena que contenga 1 bit en la posición que especifica la señal id.

Implementar el módulo propuesto usando el concepto de decodificador.

Tabla de verdad

EN	w2	w1	w0	F7	F6	F5	F4	F3	F2	F1	F0	OPERACIÓN
1	0	0	0	0	0	0	0	0	0	0	1	Girar a la derecha
1	0	0	1	0	0	0	0	0	0	1	0	Girar a la izquierda
1	0	1	0	0	0	0	0	0	1	0	0	Avanzar
1	0	1	1	0	0	0	0	1	0	0	0	Retroceder
1	1	0	0	0	0	0	1	0	0	0	0	Detenerse
1	1	0	1	0	0	1	0	0	0	0	0	Elevar Manos
1	1	1	0	0	1	0	0	0	0	0	0	Bajar manos
1	1	1	1	1	0	0	0	0	0	0	0	Movimiento extra
0	X	X	X	0	0	0	0	0	0	0	0	Nada

Implementación Robot UTP

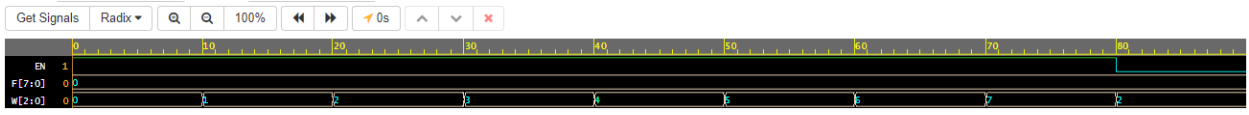
```
module Robot(
input logic EN,
input logic [2:0]W,
output logic [7:0]F);

always@(*)
  begin
    if (EN==1)
      begin
        case (W)
        3'b000:
           F= 8'b00000001;
        3'b001:
           F= 8'b00000010;
        3'b010:
           F= 8'b00000100;
        3'b011:
           F= 8'b00001000;
```

```
          3'b100:
             F= 8'b00010000;
          3'b101:
             F= 8'b00100000;
          3'b110:
             F= 8'b01000000;
          3'b111:
             F= 8'b10000000;
        endcase
        end
      F= 8'b00000000;
    end
endmodule
```

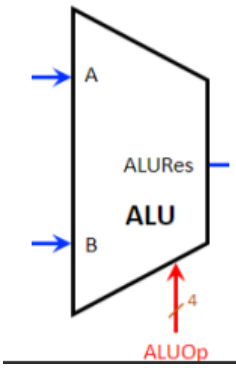
Testebench Robot UTP

```
module Testbench;
 logic EN =1;
 logic [2:0]W;
 logic [7:0]F;

 Robot robot (EN,W,F);

   initial begin
     $dumpfile ("dump.vcd");
     $dumpvars(1);

         W= 3'b000;
     #10 W= 3'b001;
     #10 W= 3'b010;
     #10 W= 3'b011;
     #10 W= 3'b100;
     #10 W= 3'b101;
     #10 W= 3'b110;
     #10 W= 3'b111;
     #10 EN= 0;
     W= 3'b010;
     #10
     $finish;
   end
endmodule
```

Simulación Robot UTP



Unidad aritmeticológica (ALU).



ALUOp	operación
0001	Suma (+)
0010	Resta (-)
0011	And (&)
0100	Or (|)
0101	Xor (^)
0110	Corrimiento lógico a la izquierda (<<)

Creación de una ALU de dos operandos (A y B tamaño de 32 bits), un ALUOp (tamaño de 4 bits) que asigna una operación y devuelve el resultado (tamaño de 32 bits) DIseño ALU

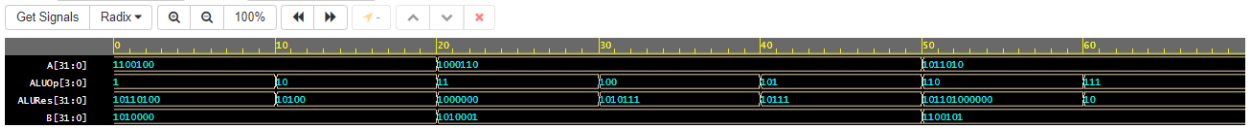
```
module ALU (input logic [31:0]A,
            input logic [31:0]B,
            input logic [3:0]ALUOp,
            output logic [31:0]ALURes);

  always@ (*)
    begin
      case (ALUOp)
      4'b0001:
        ALURes <= A+B;
      4'b0010:
        ALURes <= A-B;
      4'b0011:
        ALURes <= A&B;
      4'b0100:
        ALURes <= A|B;
      4'b0101:
        ALURes <= A^B;
      4'b0110:
      ALURes <= A<< B[4:0];
      4'b0111:
      ALURes <= A>> B[4:0];
    endcase
    end
endmodule
```

Tesbench ALU

```

module test;
  logic [31:0]A;
  logic [31:0]B;
  logic [3:0]ALUOp;
  logic [31:0]ALURes;

         ALU alu (A,B,ALUOp,ALURes);

```

```
        initial begin
          $dumpfile ("dump.vcd");
          $dumpvars(1);

       ALUOp= 4'b0001; A= 32'b1100100; B= 32'b1010000;
   #10 ALUOp= 4'b0010; A= 32'b1100100; B= 32'b1010000;
   #10 ALUOp= 4'b0011; A= 32'b1000110; B= 32'b1010001;
   #10 ALUOp= 4'b0100; A= 32'b1000110; B= 32'b1010001;
   #10 ALUOp= 4'b0101; A= 32'b1000110; B= 32'b1010001;
   #10 ALUOp= 4'b0110; A= 32'b1011010; B= 32'b1100101;
   #10 ALUOp= 4'b0111; A= 32'b1011010; B= 32'b1100101;
   #10
          $finish;
        end
    endmodule
```

Simulación ALU



Procedimiento

Ejercicio 1: Tabla de verdad 4 a 1

-Se elabora el mapa de Karnaugh de 4 variables y se realizan las agrupaciones de 1.
-Con la ecuación Booleana se realiza la implementación en System Verilog y adicionalmente se elaboran los llamados a compuertas lógicas para la segunda implementación.
-Finalmente se realiza el Testbench con todos los posibles valores que pueden tomar las variables de entrada.

Ejercicio 2: Función lógica 5 a 1

-Se elaboran las tablas de verdad siendo S la salida de la función, asignando un 1 para los valores que están en la función y 0 para los valores que no están en la función.
-Se elabora el Mapa de Karnaugh de 5 variables y posteriormente se escribe la ecuación Booleana resultante.
-Se realiza la implementación tanto en ecuación Booleana como en llamados a compuertas lógicas A continuación se elabora el Testbench para los posibles valores de 5 variables de entrada.

Ejercicio 3: Circuito 8 a 1

-Se definen las salidas de cada compuerta (Los cables) que van a servir como entradas de las siguientes compuertas.
-Se hace la ecuación Booleana correspondiente al circuito y para el llamado a módulos de compuertas lógicas se descompone la compuerta XNOR en compuertas básicas OR y AND.
-Se realiza el Testbench para los valores de las 8 entradas.

Ejercicio 4: Bombilla 3 a 1

-Se hace la tabla de verdad con las tres variables y una salida S, asignando 1 como salida

cuando la bombilla enciente y un 0 cuando no hay conexión.
-Se elabora la ecuación booleana y finalmente se hace la implementación y el Testbench correspondiente.

Ejercicio 5: Robot UTP
-Se realiza la tabla de verdad teniendo en cuenta la operación que se piensa realizar, el enable y las entradas.
-Se establece los casos para cada acción que puede realizar el robot.
-Se realiza el Testbench con los valores de prueba.

Ejercicio 6: ALU
-Se declaran ambas entradas de 32 bits, la respuesta de 32 bits y el ALUOp de 4 bits.
-Se realizan los posibles casos para cada operación entre las variables de entrada A y B.
-Se implementa el Testbench para los valores de prueba.

Interpretación resultados

Primer ejercicio: Los valores en la simulación son los esperados con respecto a la tabla de verdad.

Segundo ejercicio: Los valores en la simulación son los esperados según la función lógica y sus salidas.

Tercer ejercicio: Los valores en la simulación son los esperados según el circuito de 8 entradas y su tabla de verdad correspondiente.

Cuarto ejercicio: Los valores en la simulación son los esperados en relación a los posibles casos en los que se prende la bombilla.

Quinto ejercicio: Los valores de la simulación son los esperados según el módulo de prueba para el robot UTP.

Sexto ejercicio: Los resultados en la simulación son los esperados según las entradas que se le asignan a la ALU y sus operaciones entre sí.

Conclusiones
Tener conocimiento sobre lógica combinatoria y expresiones Booleanas es fundamental para realizar proyecto con más complejidad.

Problemas encontrados:
No se encontraron problemas durante la elaboración de este laboratorio.